8. 非線形計画法

1. 目的

本実験の目的は, システム工学の重要な 1 分野である非線形計画法について実際にコンピュータを使って問題を解くことにより理解を深めることである。

2. 原理

2.1 制約なし最適化問題の解法

本項では, 制約条件がない状態で非線形関数を最適化する (最大値あるいは最小値を求める) ための代表的なアルゴリズムである最急降下法と Newton 法について述べ, これらの対照的な性質について説明する。さらに, 最急降下法や Newton 法を含むアルゴリズムの一般形について触れる。

なお, これ以降では, 太字で行ベクトルや列ベクトルをあらわし, 下付き添字 ($\boldsymbol{x}_k$ など) でベクトルやスカラーの第 k 回目の繰り返しにおける値をあらわす。ベクトル $\boldsymbol{x}$ の第 k 成分をあらわすときには, x の部分を太字にせずに「x_k」と書く。 これらの記法に混乱しないよう注意すること。また, 記号 $\mathbf{0}$ によって零ベクトルをあらわし, 数 0 と区別する。なお, ベクトルは特に断らない限りは縦ベクトルであるものとする。

2.1.1 最急降下法

$\boldsymbol{x}$ を n 次のベクトル, f をスカラー関数とし,

$$y = f(\boldsymbol{x}) \tag{8.1}$$

が最小値を取る $\boldsymbol{x}$ を計算機によって求める問題を考える。ただし, f は必要な回数だけ微分可能であると仮定しておく。この問題を解くための最も簡単な方法が, 最急降下法あるいは勾配法と呼ばれる方法である。

f の勾配ベクトル

$$\nabla f = \left[\frac{\partial f}{\partial x_1}, \ldots, \frac{\partial f}{\partial x_n}\right] \tag{8.2}$$

は, 関数 $f(\boldsymbol{x})$ の「等高面」, すなわち

$$f(\boldsymbol{x}) = \text{一定} \tag{8.3}$$

という n 次元空間内の曲面の法線ベクトルを与える。このベクトルの向きは関数の値の増加する方向になる。

例 1 関数

$$f(x_1, x_2) = x_1^2 + x_2^2 \tag{8.4}$$

を考える。式 (8.4) の等高線は円である。一方,

$$\frac{\partial f}{\partial x_1} = 2x_1, \quad \frac{\partial f}{\partial x_2} = 2x_2 \tag{8.5}$$

であるから, ある点における勾配ベクトルの向きは原点とその点を結ぶ方向になる。これは円の法線ベクトルである (図 8.1)。

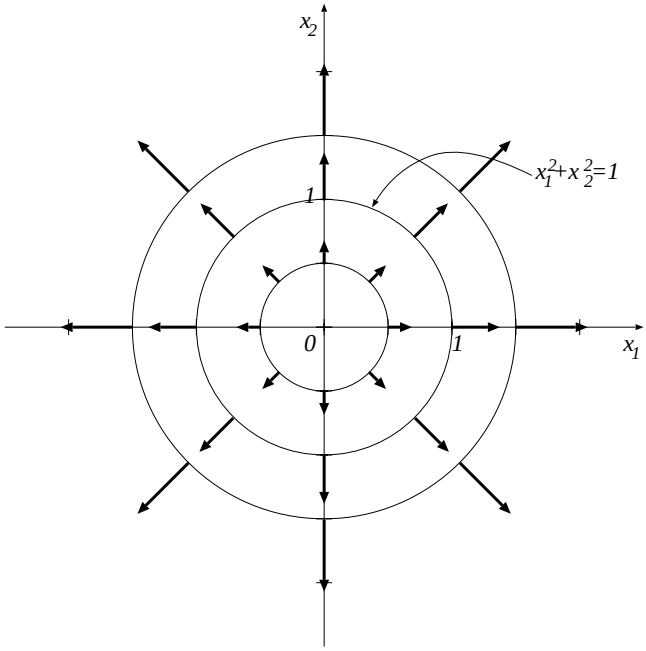

図 8.1 関数 $f(x_1, x_2) = x_1^2 + x_2^2$ の等高線と勾配ベクトル

例 2 関数

$$f(x_1, x_2) = x_1 + x_2 \tag{8.6}$$

を考える。式 (8.6) の等高線は傾き -1 の直線である。一方,

$$\frac{\partial f}{\partial x_1} = 1, \quad \frac{\partial f}{\partial x_2} = 1 \tag{8.7}$$

であるから, ある点における勾配ベクトルはつねにベクトル $[1, 1]^T$ に平行になる。これは傾き -1 の直線と直交する (図 8.2)。

さて, 図 8.1 のように, 式 (8.1) の関数が極大値を取らず, かつ唯一の点において極小値を取る場合を考えよう。極小値が唯一であることからこの値は最小値にもなるわけだが, この場合, 等高線の法線ベクトルの逆向きの方向は関数の値が一番減る方向なのだから, 法線ベクトルの逆向きに解の候補を少しずつ動かしてゆ

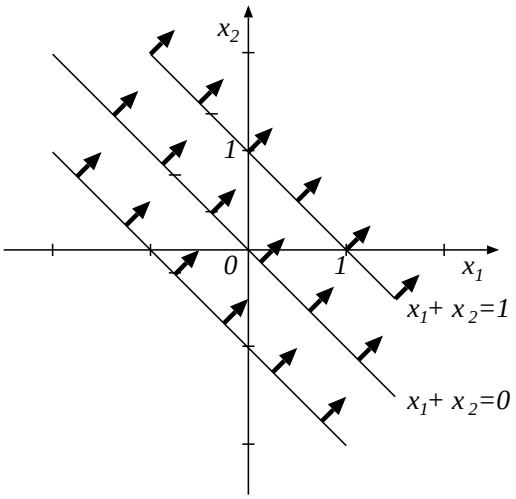

図 8.2 関数 $f(x_1, x_2) = x_1 + x_2$ の等高線と勾配ベクトル

く以下のような繰り返し計算をおこなうと, 繰り返しの回数を十分多く取れば, この関数の最小値を (もしあれば) 求めることができると期待できる。

G1) 初期値 $\boldsymbol{x}_0$ を適当に定め, $k = 0$ とする。

G2) $\nabla f(\boldsymbol{x}_k) = \boldsymbol{0}^T$ なら終了。

G3) 探索ベクトル $\boldsymbol{d}_k$ を

$$\boldsymbol{d}_k = -(\nabla f(\boldsymbol{x}_k))^T \tag{8.8}$$

によって定め, ステップ幅 α_k を適当に決め,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k \tag{8.9}$$

により $\boldsymbol{x}_{k+1}$ を定める。

G4) $k = k + 1$ とする。

このような方法を最急降下法と呼ぶ。

なお, ステップ幅 α_k は, アルゴリズムの安定性という観点からは小さい正の定数であれば良いのだが, このようにすると収束が極めて遅くなり実用に耐えない。だから, 最急降下法を用いて実用的なプログラムを作成するときには, 直線探索と呼ばれる方法を用いてステップ幅 α_k を決定することが多い。ただし直線探索についてはこの実験では取り扱わない。

最急降下法には, 簡単である, 各ステップごとの計算量が少ない, 関数が唯一の極小点を持つときには大域的な収束性が保証されるという利点がある一方で, 解の収束が極めて遅いという欠点がある。また, 関数に複数の極小点があるときには, 求められるのはいくつかの極小点のうちのいずれかであって, それが最小点であるという保証はない。

2.1.2 Newton 法

最急降下法が持つ収束の遅さという欠点を克服する方法のひとつに Newton 法がある。

再び式 (8.1) の最小点を求める問題を考える。式 (8.1) を $\boldsymbol{x}_k$ のまわりで Taylor 展開して 2 次の項まで残すと,

$$\begin{aligned} f(\boldsymbol{x}) \simeq\ & f(\boldsymbol{x}_k) + \nabla f(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) \\ & + \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}_k)^T \nabla^2 f(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) \end{aligned} \tag{8.10}$$

が得られる。 ここに,

$$\nabla^2 f = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1 \partial x_1} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \vdots & & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \cdots & \dfrac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix} \tag{8.11}$$

である。 $\nabla^2 f$ は f の Hesse 行列と呼ばれる。 f が 2 階連続微分可能であれば $\nabla^2 f$ は対称行列になる。

さて, 式 (8.10) の近似の精度が十分に良ければ, 式 (8.10) の右辺が最小になる $\boldsymbol{x}$ を求めることで, 最小点の良い近似が得られると期待される。 ところで, $\boldsymbol{x}^*$ が関数 f の極小点であれば f の Hesse 行列は正定値となることが証明できる。上記は「スカラー関数がある点で極小値を取る (下に凸になる) ときにはその関数の 2 回微分は正になる」という事実の n 次元への拡張になっている。

記法の簡単のために $H = \nabla^2 f(\boldsymbol{x}_k)$, $\boldsymbol{p} = (\nabla f(\boldsymbol{x}_k))^T$, $\boldsymbol{\xi} = \boldsymbol{x} - \boldsymbol{x}_k$ とおいて式 (8.12) の右辺を書き直すと

$$f(\boldsymbol{x}_k) + \boldsymbol{p}^T \boldsymbol{\xi} + \frac{1}{2}\boldsymbol{\xi}^T H \boldsymbol{\xi} \tag{8.12}$$

となる。式 (8.12) を

$$f(\boldsymbol{x}_k) - \frac{1}{2}\boldsymbol{p}^T H^{-1} \boldsymbol{p} + \frac{1}{2}(\boldsymbol{\xi} + H^{-1}\boldsymbol{p})^T H (\boldsymbol{\xi} + H^{-1}\boldsymbol{p}) \tag{8.13}$$

のように書き直し, H が正定値であることに注意すると, 式 (8.13) は

$$\boldsymbol{\xi} = -H^{-1}\boldsymbol{p} \tag{8.14}$$

のときに最小値を取ることがわかる。式 (8.14) をもとの記号を使って書き直して

$$\boldsymbol{d}_k = -(\nabla^2 f(\boldsymbol{x}_k))^{-1} (\nabla f(\boldsymbol{x}_k))^T \tag{8.15}$$

と定義し, $\boldsymbol{\xi} + \boldsymbol{x}_k$ を改めて $\boldsymbol{x}_{k+1}$ とおくと,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \boldsymbol{d}_k \tag{8.16}$$

という式が得られる。

最適化したい関数が 2 次である場合には式 (8.16) が最も良い $\boldsymbol{x}_{k+1}$ を与えるのであるが, 関数が 2 次でない場合には $\boldsymbol{x}_{k+1}$ をベクトル $\boldsymbol{d}_k$ の方向に 1 以外の大きさで動かした方が良い結果を与えることもある。 そのような場合に対処するためには, ステップ幅 α_k を適当に定めて,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k \tag{8.17}$$

とする。

以上により,

N1) 適当に初期値 $\boldsymbol{x}_0$ を定め, $k = 0$ とする

N2) $\nabla f(\boldsymbol{x}_k) = \boldsymbol{0}^T$ なら終了

N3) 線形方程式

$$\nabla^2 f(\boldsymbol{x}_k)\boldsymbol{d}_k = -(\nabla f(\boldsymbol{x}_k))^T \tag{8.18}$$

を解いて探索ベクトル $\boldsymbol{d}_k$ を求める

N4) ステップ幅 α_k を適当に定めて

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k \tag{8.19}$$

により解を更新する

N5) $k = k + 1$ とする

という Newton 法のアルゴリズムが得られる。

Newton 法は一般に最急降下法と比べて極めて高速であり, 特に関数 f が 2 次のときには 1 回の反復で最適解が得られるという利点がある一方で, 初期値が十分最適値に近くないと収束性が保証されない, $\nabla^2 f$ が正定値でないと利用できない, $\nabla^2 f$ が正確に求められていないと不安定になる, $\nabla^2 f$ の計算に手間がかかる, という欠点がある。また, 最急降下法と異なり, Newton 法では, ステップ幅 α_k を 1 に固定しても良い収束特性が得られることも多い。なお, 関数に複数の極小点があるときには求められるのは極小点のどれかであって最小点とは限らないことは最急降下法と同じである。

Newton 法の変種に, $\nabla^2 f$ を直接計算するのではなく, $\boldsymbol{x}_k$ の更新と並行して逐次的に近似してゆくアルゴリズムもある。このような一連のアルゴリズムを準 Newton 法と呼ぶ。

2.1.3 最適化アルゴリズムの一般形

最適化アルゴリズムは

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \alpha_k H_k \nabla f(\boldsymbol{x}_k) \tag{8.20}$$

という形を取ることが多い。 ここに, H_k は k に依存する n 行 n 列の行列である。式 (8.20) において H_k を単位行列とすると最急降下法が得られ, $H_k = (\nabla^2 f(\boldsymbol{x}_k))^{-1}$ とすると Newton 法が得られる。

2.2 制約付き最適化問題と Lagrange 関数

前の項では変数に制約条件がないときの関数の最適化問題を取り扱った。本節では, 変数に等式制約条件が課されているときの最適化問題の解法を取り扱う。 このような問題の代表的な解法が Lagrange 関数を用いる方法である。

等式制約条件

$$g(\boldsymbol{x}) = 0 \tag{8.21}$$

のもとで関数 $f(\boldsymbol{x})$ の最小値を求める問題を考える。ただし, $g(\boldsymbol{x})$ は必要な回数だけ微分可能であるとする。

ここで, 点 $\overline{\boldsymbol{x}}$ が上記の問題の最適解だったと仮定しよう。すると, $g(\boldsymbol{x}) = 0$ なる曲面に沿って $\overline{\boldsymbol{x}}$ を少しだけ動かしても, $f(\boldsymbol{x})$ の値は変わらないはずである。 すなわち, 超曲面 $f(\boldsymbol{x}) = f(\overline{\boldsymbol{x}})$ と超曲面 $g(\boldsymbol{x}) = 0$ は点 $\overline{\boldsymbol{x}}$ において接 (超) 平面を共有するから, これらの超曲面の法線ベクトル ∇f と ∇g は点 $\overline{\boldsymbol{x}}$ において平行になる。このような場合には, 適当な定数 λ が取れて,

$$\nabla f(\overline{\boldsymbol{x}}) + \lambda \nabla g(\overline{\boldsymbol{x}}) = 0 \tag{8.22}$$

となる。

式 (8.22) の左辺は

$$L(\lambda, \boldsymbol{x}) = f(\boldsymbol{x}) + \lambda g(\boldsymbol{x}) \tag{8.23}$$

なる関数の $\boldsymbol{x}$ に関する偏微分になっている。 だから, 点 $\overline{\boldsymbol{x}}$ が最適であるための必要条件 (8.22) は,

$$\frac{\partial L}{\partial \boldsymbol{x}} = \boldsymbol{0}^T \tag{8.24}$$

と書き直される。 一方, 式 (8.23) を λ について偏微分すると

$$\frac{\partial L}{\partial \lambda} = g \tag{8.25}$$

となる。最適解 $\overline{\boldsymbol{x}}$ には $g(\overline{\boldsymbol{x}}) = 0$ という制約条件が課されていたのだが, この制約条件は, 式 (8.23) の関数 L を用いることで,

$$\frac{\partial L}{\partial \lambda} = 0 \tag{8.26}$$

と書き直されることになる。

式 (8.24) と式 (8.26) をまとめると, 点 $\overline{\boldsymbol{x}}$ が最適解であるための必要条件は,

$$\frac{\partial L}{\partial \boldsymbol{x}} = \boldsymbol{0}^T, \quad \frac{\partial L}{\partial \lambda} = 0 \tag{8.27}$$

であることがわかる。 よって, この制約付き最小化問題は, 非線形連立方程式 (8.27) を解く問題に帰着されることになる。

ただし, 式 (8.27) は最小解のための必要条件であって十分条件ではない。実際には, 式 (8.27) を解いて得られる解には関数の極小点, 極大点および鞍点がすべて含まれる。

例 3 関数

$$f(x_1, x_2) = x_1 + x_2, \quad g(x_1, x_2) = x_1^2 + x_2^2 \quad (8.28)$$

に対し, 制約条件 $g(x_1, x_2) = 1$ のもとで $f(x_1, x_2)$ を最小化する問題を考えよう。制約条件を満たす領域は単位円である。一方, 関数 f の等高線は傾き -1 の直線である。

さて, 単位円上にある点が円周上のある点を起点として微小に移動するという状況を考える。

点 (x_1, x_2) が動き始める点の座標が $(1/\sqrt{2}, 1/\sqrt{2})$ である場合と $(-1/\sqrt{2}, -1/\sqrt{2})$ である場合には, この点から点 (x_1, x_2) が少しだけ動いたとき, その軌跡は $f(x_1, x_2)$ の等高線を横切らないから, $f(x_1, x_2)$ の値は変わらない。そして, 点 $(1/\sqrt{2}, 1/\sqrt{2})$ における $g(x_1, x_2) = 1$ を満たす曲線 (円) の法線ベクトルは $[\sqrt{2}, \sqrt{2}]^T$, 点 $(-1/\sqrt{2}, -1/\sqrt{2})$ におけるの法線ベクトルは $[-\sqrt{2}, -\sqrt{2}]^T$ となり, これらはいずれも $f(x_1, x_2)$ の等高線に直交している (例 1, 例 2 参照)。

これに対し, 点 (x_1, x_2) が動き始める点の座標が上記以外であった場合には, 点が微小に移動したときに, その軌跡は必ず $f(x_1, x_2)$ の等高線を横切るので, 関数の値は変動する。また, このとき, $g(x_1, x_2) = 1$ を満たす曲線の法線ベクトルは $f(x_1, x_2)$ の等高線と直交しない。

以上のように考えると, Lagrange 乗数法によって関数 $f(x_1, x_2)$ が極値を取る点 $(1/\sqrt{2}, 1/\sqrt{2})$ および $(-1/\sqrt{2}, -1/\sqrt{2})$ が発見されることがわかる。なお, 前者は関数が極大となる点, 後者は関数が極小となる点である。

単位円上の各点 ($g(x_1, x_2) = 0$ をみたす点) における法線ベクトルと関数 $f(x_1, x_2)$ の勾配ベクトルとの関係を図 8.3 に示す。図 8.3 では, 点 A および点 B において単位円の法線ベクトルと関数 $f(x_1, x_2)$ の勾配ベクトルが平行になっている。点 A が関数 $f(x_1, x_2)$ が最小となる点であり, 点 B が関数 $f(x_1, x_2)$ が最大となる点である。点 C および点 D では, 単位円の法線ベクトルと関数 $f(x_1, x_2)$ の勾配ベクトルは斜めに交わっている。このとき, 点 C および点 D から出発した点の軌跡は関数 $f(x_1, x_2)$ の等高線を必ず横切るから, 点 C および点 D では関数 $f(x_1, x_2)$ は極値を取らないことがわかる.

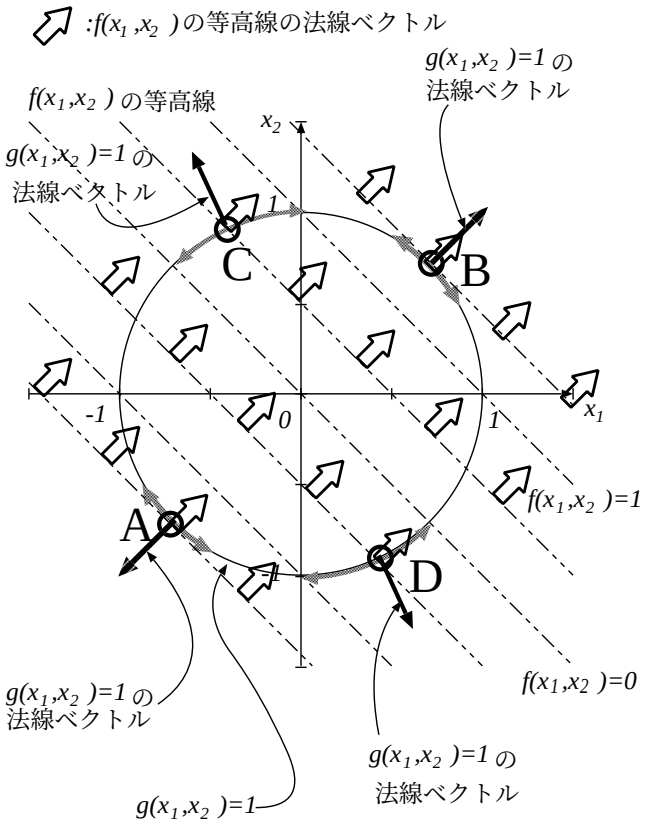

図 8.3 Lagrange 乗数法の意味

制約条件が $g_1(\boldsymbol{x}) = 0$ から $g_m(\boldsymbol{x}) = 0$ まで m 個あるときにも,

$$L(\lambda, \boldsymbol{x}) = f(\boldsymbol{x}) + \sum_{i=1}^{m} \lambda_i g_i(\boldsymbol{x}) \quad (8.29)$$

とおき,

$$\frac{\partial L}{\partial \boldsymbol{x}} = \mathbf{0}^T, \frac{\partial L}{\partial \lambda_1} = 0, \ldots \frac{\partial L}{\partial \lambda_m} = 0 \quad (8.30)$$

なる非線形連立方程式を解くことにより, もとの最適化問題を解くことができる。

式 (8.23), 式 (8.29) で定義される L を Lagrange 関数, 式 (8.27), 式 (8.30) を解くことによりもとの関数の最小値を得る解法を Lagrange 乗数法と呼ぶ。

続いて, 式 (8.30) の解を求める方法を導くために, 非線形方程式

$$\boldsymbol{r}(\boldsymbol{x}) = \mathbf{0} \quad (8.31)$$

を解く問題を考える。 ここに, $\boldsymbol{r}$ は n 次のベクトル値関数とする。

k 回の繰り返し時における解の候補 $\boldsymbol{x}_k$ が与えられているとしよう。式 (8.31) を $\boldsymbol{x}_k$ のまわりで Taylor 展開して 1 次の項まで取ると,

$$\boldsymbol{r}(\boldsymbol{x}) \simeq \boldsymbol{r}(\boldsymbol{x}_k) + \nabla \boldsymbol{r}(\boldsymbol{x}_k)(\boldsymbol{x} - \boldsymbol{x}_k) \tag{8.32}$$

となるから[1] , $\boldsymbol{r}(\boldsymbol{x})$ を $\mathbf{0}$ に近くするためには,

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - (\nabla \boldsymbol{r}(\boldsymbol{x}_k))^{-1} \boldsymbol{r}(\boldsymbol{x}_k) \tag{8.33}$$

とすればよいことがわかる。 ただし, 数値的には逆行列を求めることは好ましくないので, $\boldsymbol{x}_{k+1}$ を式 (8.33) から直接計算するかわりに線形方程式

$$(\nabla \boldsymbol{r}(\boldsymbol{x}_k))\boldsymbol{x}_{k+1} = (\nabla \boldsymbol{r}(\boldsymbol{x}_k))\boldsymbol{x}_k - \boldsymbol{r}(\boldsymbol{x}_k) \tag{8.34}$$

を解く。 上記の手順を繰り返すことで, 式 (8.31) の解が求められると考えられる。

式 (8.33) の手順を繰り返すことによって関数の零点を求めるアルゴリズムも Newton 法と呼ばれる。

式 (8.27) あるいは式 (8.30) を解く問題に上述のアルゴリズムを適用するためには, 変数 $\boldsymbol{x}$ に関する非線形連立方程式 $\boldsymbol{r}(\boldsymbol{x}) = 0$ を変数 $(\boldsymbol{x}, \lambda)$ に関する非線形連立方程式 (8.27) あるいは変数 $(\boldsymbol{x}, \lambda_1, \ldots, \lambda_m)$ に関する非線形連立方程式 (8.30) で置き換えればよい。

3. 実験

3.1 実験課題

課題 1 教官が与えた制約条件なし最適化問題を最急降下法および Newton 法を用いて解くプログラムを作成せよ。

課題 2 教官が与えた等式制約付き最適化問題を Lagrange 乗数法を用いて解くプログラムを作成せよ。

3.2 実験方法

Mule を用いてプログラムを作成し, Scilab で実行する。

Scilab の使い方についてはすでに基礎実験で学んでいるはずなので, ここではこく簡単な事項のみ復習しておく。各自で必要に応じてオンラインマニュアルや端末室にある資料などを参照すること。

Scilab の起動 Scilab を起動するには,

```
scilab↩
```

と入力する。

スクリプトファイルの実行 スクリプトファイルを実行するときには, exec という関数を使う。

たとえば sample.sci というファイルを実行したいときには, Scilab のウィンドウ内で

```
exec("sample.sci");↩
```

と入力する。ただし, 上記の書き方では, カレントディレクトリにあるファイルしか読めない。ディレクトリを指定したいときには,

```
exec("/home/b/y99/d99999/sample.sci");↩
```

などのようにパスを付けてファイル名を指定する。

変数をファイルに保存する 変数をファイルに保存したいときには, write という関数を使う。

たとえば w という変数を w.txt というファイルに保存したいときには,

```
write("w.txt",w)↩
```

などと入力する。

現在作業中のディレクトリにこれから保存しようとしているファイルと同一の名前のファイルがあると, 関数 write は変数の保存に失敗する。だから, 関数 write を使う前に, 必要に応じて同名のファイルを消去しておかなければならない。

グラフを PostScript 形式で保存する Scilab のグラフィックスウィンドウの左上の File メニューから Export を選択する (図 8.4)。すると図 8.5 のようなウィンドウがあらわれるので,

1. 一番上の Format Selection という欄で Postscript という部分を選択する (その部分にマウスカーソルを合わせてマウス左ボタンを押す)
2. 上から 2 番目の Type という欄では blackwhite を選択する
3. 上から 3 番目の Orientation という欄で portrait を選択する
4. 上から 4 番目の Filename without extension という部分にはファイル名を拡張子なしで指定する (.eps という拡張子は自動的に生成される)
5. ウィンドウ左下の Ok という部分にマウスカーソルを合わせてマウス左ボタンを押す

という操作をおこなう。すると, 画像が指定したファイルに保存される。図 8.5 では test.eps というファイルを作成しようとしている。

[1] 記号 $\nabla \boldsymbol{r}$ は $\begin{bmatrix} \frac{\partial r_1}{\partial x_1} & \cdots & \frac{\partial r_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial r_n}{\partial x_1} & \cdots & \frac{\partial r_n}{\partial x_n} \end{bmatrix}$ という行列をあらわす。

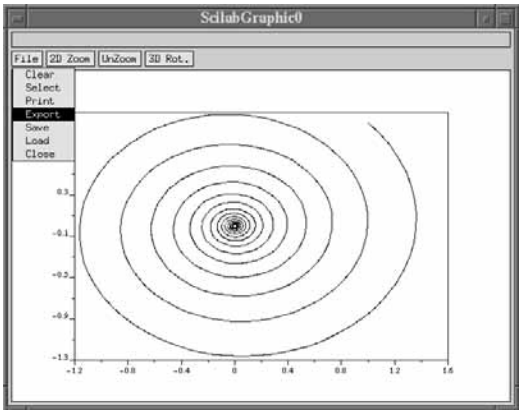

図 8.4 グラフを PostScript 形式で保存 (1)

図 8.5 グラフを PostScript 形式で保存 (2)

Scilab を終了する　Scilab を終了したいときには,

```
quit↩
```

と入力する。

参考プログラム　参考のために, いくつか Scilab のプログラムの例を挙げておく。

例 4 オイラー法を用いて 2 次元数空間における微分方程式

$$\frac{d}{dt}\boldsymbol{x} = \begin{bmatrix} -0.1 & 1 \\ -1 & -0.1 \end{bmatrix} \boldsymbol{x} \tag{8.35}$$

を数値的に解き, 解曲線を 2 次元空間にプロットすることを考える。ただし, $\boldsymbol{x}$ の初期値を $[1,1]^T$ とする。

周知のように, オイラー法は, 微分方程式 (8.35) を

$$\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + h \begin{bmatrix} -0.1 & 1 \\ -1 & -0.1 \end{bmatrix} \boldsymbol{x}_k \tag{8.36}$$

のように近似する方法である。ここでは $h = 0.1$ と取ることにする。

繰り返しの計算には while 文を用いることにする。また, if 文および break 文の使い方の例を示すために, やや冗長なのだが,

- while 文自体は無限ループにする
- 変数 i を 0 で初期化し, ループ 1 回ごとにこの変数の値を 1 ずつ増やす
- 変数 i が 1000 を越えたら終了する

という制御構造を採用する。さらに, 軌跡の描画のために, 解の履歴を w という変数に保存しておくことにする。

このためのプログラムは以下の通りである。

```
A=[-0.1,1;-1,-0.1]; //行列 A の定義
x=[1;1]; //x の初期値の定義
w=x;     //過去の履歴を保存する変数
i=0;     //終了判定用変数
h=0.1;   //ステップ幅
while 1  //無限ループ
        x=x+h*A*x; //オイラー法
        w=[w,x];   //履歴を保存
        i=i+1; //終了判定用変数を増加
        if i>1000 then //終了判定
                break; //ループ脱出
        end //if 文終了
end  //while 文終了
plot(w(1,:),w(2,:)) //解の軌跡を描画
```

このプログラムを実行して得られる結果が, 先の図 8.4 に挙げたものである。

なお, Scilab のスクリプトにおいて, 文字//およびそれ以降は注釈とみなされる。注釈は, 人間にとってスクリプトを読みやすくする目的で書かれるもので, プログラム本体とはまったく関係はない。

例 5 関数

$$f(x_1, x_2) = x^6 + x^2y^2 + y^6 \tag{8.37}$$

の勾配ベクトルは

$$\nabla f = \begin{bmatrix} 6x^5 + 2xy^2 & 2x^2y + 6y^5 \end{bmatrix} \tag{8.38}$$

であり, Hesse 行列は

$$\nabla^2 f = \begin{bmatrix} 30x^4 + 2y^2 & 4xy \\ 4xy & 2x^2 + 30y^4 \end{bmatrix} \tag{8.39}$$

なのであるが, 引数 x が与えられたときにこれらを求める関数 grad と hesse を作ってみよう。ただし, 関数の定義だけが書かれたファイル fn.sci を用意することにする。

さて, ファイル fn.sci の内容は以下のようになる。

```
function g=grad(x)//勾配ベクトル
g(1)=6*x(1)**5 + 2 * x(1)* x(2)**2
g(2)=6*x(2)**5 + 2 * x(1)**2 * x(2)
g=g'

function h=hesse(x)//ヘッセ行列
h(1,1)=30 * x(1)**4 + 2* x(2)**2
h(1,2)=4  * x(1) * x(2)
h(2,1)=4  * x(1) * x(2)
h(2,2)=30 * x(2)**4 + 2* x(1)**2
```

ここで定義された関数を使いたいときには, Scilab のウィンドウ内で

```
getf('fn.sci')↩
```

と入力する。ファイルがカレントディレクトリにない場合にはパスの指定が必要になる。

他のスクリプトから上記で定義された関数 grad と関数 hesse を使いたいときには, スクリプト中のこれらの関数を使用する部分より前のところに

```
getf('fn.sci')↩
```

と書いておけばよい。

では, ファイル fn.sci で定義された関数 grad を用いて, 2 次元空間内で式 (8.38) の勾配ベクトルの正の向き (関数の値が発散する方向) に向かって点を動かしてゆくスクリプトを示そう。

```
getf('fn.sci');
x=[0.1;-0.1];
w=x;
while(norm(x)<1)
    x=x+grad(x)';
    w=[w x];
end
plot(w)
```

このスクリプトを実行すると, 図 8.6 のようなグラフが得られる。

Scilab のグラフィックス関数の使い方について　v と w を同じ長さのベクトルとする。さて, Scilab の関数 plot は,

```
plot(v,w)↩
```

と入力すると, ベクトル v の各成分を描画すべき点の横軸の座標, ベクトル w の各成分を描画すべき点の縦軸の座標とみなしてグラフの描画をおこなう。一方, M を m 行 n 列の行列としたとき, Scilab で

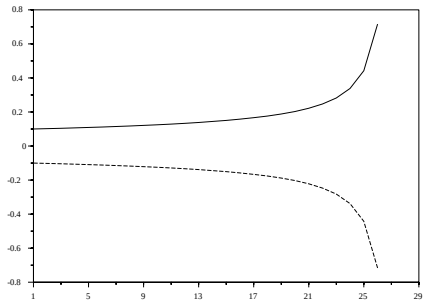

図 8.6 勾配方向に発散してゆく軌道の各座標成分

```
plot(M)↩
```

というコマンドを実行すると, Scilab は行列 M の各行ベクトルを独立したグラフに対応するデータとみなして複数のグラフを重ね書きする。この場合, グラフの横軸には 1 から始まる整数が自動的に割り振られる。

[注意事項]

- プログラムを作成する際には, プログラムの停止条件をどうするかが問題となる。理論的には勾配ベクトルが零になったら極小点が発見されたとみなしてプログラムを停止すればよいのだが, コンピュータの数値計算には誤差があるのと, 厳密な意味で極小点に到達するには無限回の反復を要することがあるので, これでは実用にならない。実用的には, 勾配ベクトルのノルム (長さ) が十分小さくなったら極小点が発見されたものとみなしてプログラムを停止するとよい。この場合, 勾配ベクトルのノルムがどの程度小さいときにプログラムを停止するかに応じて解の精度が決まる。

- 一般に, 所与の n 次の正方行列 A と n 次のベクトル $\boldsymbol{b}$ に対して線形方程式 $A\boldsymbol{x} = \boldsymbol{b}$ の解 $\boldsymbol{x}$ をコンピュータを使って数値的に求めるときには, たとえ行列 A が正則であっても, 行列 A の逆行列を利用して求解することは好ましくない。 これは, 逆行列の計算は極めて計算量が多く, かつ計算結果に誤差が集積しやすいからである。 多くの数値計算ソフトウエアでは, 逆行列を経由せずに直接線形方程式を解く手段がはじめから提供されている。 Scilab では, 線形方程式を解くための演算子は\である。

- プログラム完成時に面接をおこなう。面接で適切な回答ができなかった学生は, 他人のプログラム

をまる写ししたものとみなし, レポートの得点を 0 点とする。

- レポート本文は LaTeX を用いて作成するものとする。LaTeX の使い方の初歩については, 2001 年度版のプログラミング基礎の講義資料を端末室に置くので, これを参照すること。また, プログラミング基礎の講義資料には LaTeX の使い方に関する最低限の解説がなされているだけなので, 各自これに加えて適当な参考書を参照すること。

- レポート提出は電子メールのみで受け付ける。提出物は, 作成した LaTeX のソースコードおよびそれに対応する PostScript ファイル (dvips によって生成されたもの) である。

  レポートを送信する際には, LaTeX のソースコードと対応する PostScript ファイルをコマンド tar によってまとめて圧縮し, uuencode によってテキストデータに変換してから送信すること。

  例として, r.tex というソースファイルと r.ps という PostScript ファイルをまとめて r.tgz というファイルに圧縮し, これを r.uu というテキストファイルに変換してから電子メールで送信する手順を示す。

```
tar cfz r.tgz r.tex r.ps ↩
uuencode r.tgz r.tgz > r.uu ↩
cat r.uu | mail hanba@eee.u-ryukyu.ac.jp ↩
```

- ディレクトリ/home/b/teacher/hanba/ees2 に, レポート作成のための LaTeX のソースコードの雛型となるファイル sample.tex が用意されている。レポート作成の際にはこのファイルを必要に応じて適宜修正して利用すること。なお, このファイルにはレポート作成に必要な例などが含まれているが, 提出レポートでは実験に直接関係のない例などは削除すること。不適切な例が残っているレポートは減点の対象となる。

- 各学生が提出したレポートは, 過去に他のグループなどの学生が提出したレポートと電子的に比較される。他の学生のレポートとの類似性が著しいレポートは他人のレポートがまる写ししたものとみなされる。この場合, レポートの得点は 0 点となる。

- レポートには, 横軸に繰り返し回数を, 縦軸に最小化したい関数値を取ったグラフか, あるいはそれと同等のものを付けること。

- レポート作成の際に実験指導書の原理の項をまる写ししてはならない。レポートには, 実験指導書の非常に簡単な要約や, あるいは (もしあれば) 自分で調べた追加事項を書くべきである。レポートにおける実験の原理の記載が実験指導書の内容以上のものを含まない場合は, 原理の部分のページ数の上限を 1 ページとし, それを越えるレポートは大幅に減点する。 実験指導書にない事項が調べられているレポートについてはこの限りではない。

- レポート中における実験の手順に関する記述も最低限にとどめること。コンピュータを使ってプログラムを作成する実験では手順に曖昧さが生じる余地はほとんどないので, 手順に関する記述は数行程度の簡単なものでよい。特に, グラフ作成の手順などについては, 実験の本質とは無関係であるので, 記述する必要はない。

- 実験指導書に記載された例題は諸君の理解を助けるためのものであるから, これをレポートに記載するのは不適切である (担当教官にとっては自明である)。例題がまる写しされたレポートは大幅に減点する。

- この実験では長いレポートを書く必要はまったくない. 必要事項が記載されてさえいれば, レポートは 3 ページ程度の簡単なものでよい。レポートの分量そのものは評価の対象とはならないので注意すること。なるべく簡潔に要点のみ記述することを心掛けよ。

4. データ解析

- 最急降下法でステップ幅を変えたときの挙動の変化について調べよ。

- 最急降下法と Newton 法の挙動を比較せよ。

参考文献

[1] 今野 浩, 山下 浩: 「非線形計画法」, 日科技連 (1978)

[2] 藤田 宏, 今野 浩, 田邉 國士: 「最適化法」, 岩波書店 (1994)